Chiem MAT 115 Homework 3

For the third homework assignment, we will use the `babynames` dataset. This is a dataset of US baby names using information provided from the US Social Security Administration (SSA). As per usual, you should write your answers in RMarkdown. The due date for this assignment is **Sept. 25 at 2:30 PM**.

1. The `babynames` dataset is in its own package. Download it and load it into R. Display the first 8 rows of the dataset.

Note: the `babynames` dataset is a `tibble`, which is a different format from the traditional data frame. For our purposes here, it does not influence our use of the data.

```r
#install.packages('babynames')
library(babynames)
babynames
```

```
## # A tibble: 1,924,665 x 5
##     year sex   name          n   prop
##    <dbl> <chr> <chr>     <int>  <dbl>
##  1  1880 F     Mary       7065 0.0724
##  2  1880 F     Anna       2604 0.0267
##  3  1880 F     Emma       2003 0.0205
##  4  1880 F     Elizabeth  1939 0.0199
##  5  1880 F     Minnie     1746 0.0179
##  6  1880 F     Margaret   1578 0.0162
##  7  1880 F     Ida        1472 0.0151
##  8  1880 F     Alice      1414 0.0145
##  9  1880 F     Bertha     1320 0.0135
## 10  1880 F     Sarah      1288 0.0132
## # i 1,924,655 more rows
```

2. Use your R indexing skills to find out how popular (in terms of `prop`) your given name was in the year you were born. If your name is not in the dataset, then just pick a name that interests you for whatever reason.

```r
babynames[babynames$name == "Damien" & babynames$year == 2004, ]
```

```
## # A tibble: 2 x 5
##    year sex   name       n       prop
##   <dbl> <chr> <chr>  <int>      <dbl>
## 1  2004 F     Damien    11 0.00000545
## 2  2004 M     Damien  1955 0.000926
```

```r
#length(babynames[babynames$name == "Damien" & babynames$year == 2004, ])
```

A: for males my name had a prop of 0.00092558 and 0.00000545 for females in my birth year.

3. Subset the dataset into two separate ones by `sex`. For each sex, take the top-25 names by `n`. Do not over think this request. Duplicates of a name across years is acceptable. Make a boxplot of `n` by `name`. Beautify these plots in at least two ways. What strikes you about these plots? How are they similar and how are they different?

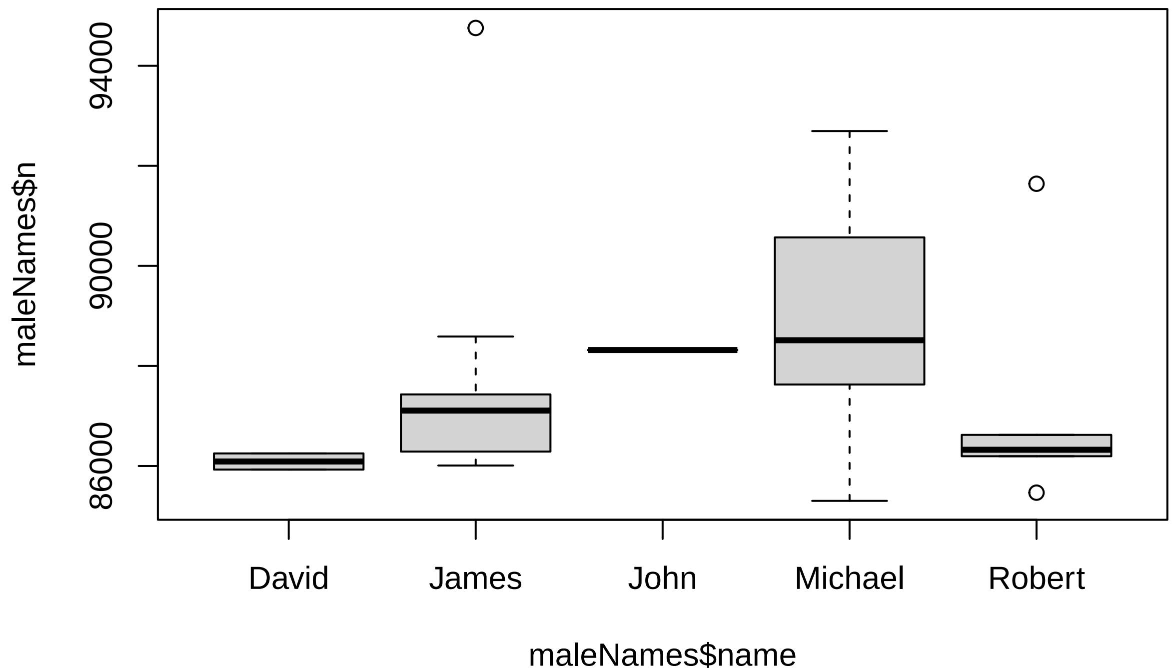
```r
maleNames <- babynames[babynames$sex == "M",]

femaleNames <- babynames[babynames$sex == "F",]

maleNames <- maleNames[order(maleNames$n, decreasing = TRUE)[1:25],]
femaleNames <- femaleNames[order(femaleNames$n, decreasing = TRUE)[1:25],]

maleNames
```

```
## # A tibble: 25 x 5
##     year sex   name        n   prop
##    <dbl> <chr> <chr>   <int>  <dbl>
##  1  1947 M     James   94756 0.0510
##  2  1957 M     Michael 92695 0.0424
##  3  1947 M     Robert  91642 0.0493
##  4  1956 M     Michael 90620 0.0423
##  5  1958 M     Michael 90520 0.0420
##  6  1948 M     James   88588 0.0497
##  7  1954 M     Michael 88514 0.0428
##  8  1955 M     Michael 88335 0.0423
##  9  1947 M     John    88318 0.0475
## 10  1946 M     James   87431 0.0530
## # i 15 more rows
```

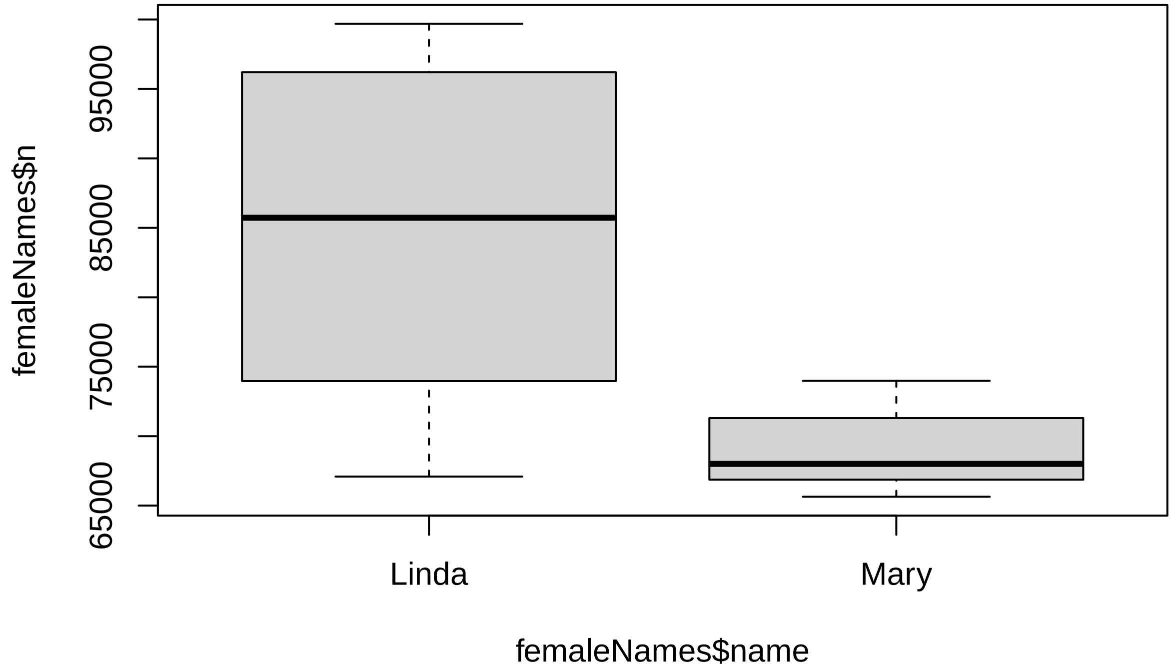
```r
femaleNames
```

```
## # A tibble: 25 x 5
##     year sex   name      n   prop
##    <dbl> <chr> <chr> <int>  <dbl>
##  1  1947 F     Linda 99686 0.0548
##  2  1948 F     Linda 96209 0.0552
##  3  1949 F     Linda 91016 0.0518
##  4  1950 F     Linda 80432 0.0457
##  5  1921 F     Mary  73982 0.0578
##  6  1951 F     Linda 73972 0.0400
##  7  1924 F     Mary  73532 0.0568
##  8  1922 F     Mary  72175 0.0579
##  9  1947 F     Mary  71688 0.0394
## 10  1923 F     Mary  71635 0.0572
## # i 15 more rows
```

```r
boxplot(maleNames$n ~ maleNames$name)
```




```r
boxplot(femaleNames$n ~ femaleNames$name)
```



4. For the entire `babynames` dataset, write a `for` loop that calculates the mean `prop` for each `year`. In your own words, explain what the output, sequence, and body of your code are doing.

```
propM <- vector(length=nrow(babynames[duplicated(babynames$year) == FALSE,]))
each_year <- vector(length=nrow(babynames[duplicated(babynames$year) == FALSE,]))
years <- babynames[duplicated(babynames$year) == FALSE, ]

for(i in 1:length(propM)){
  tempMatrix <- babynames[babynames$year == years$year[i],]

  propM[i] <- mean(tempMatrix$prop)
  each_year[i] <- tempMatrix$year[1]
}

#system.time(for(i in 1:length(propM)){
  #tempMatrix <- babynames[babynames$year == years$year[i],]

  #propM[i] <- mean(tempMatrix$prop)
  #each_year[i] <- tempMatrix$year[1]
#}) # 2.14 seconds

#propM
#each_year

propM_and_year <- data.frame(year = each_year, mean_prop = propM)
propM_and_year
```

```
##      year    mean_prop
## 1    1880 9.327296e-04
## 2    1881 9.615283e-04
## 3    1882 8.762491e-04
## 4    1883 8.952682e-04
## 5    1884 8.141704e-04
## 6    1885 8.139021e-04
## 7    1886 7.820985e-04
## 8    1887 7.867618e-04
## 9    1888 7.065885e-04
## 10   1889 7.226397e-04
## 11   1890 6.944554e-04
## 12   1891 7.028995e-04
## 13   1892 6.410328e-04
## 14   1893 6.616759e-04
## 15   1894 6.363961e-04
## 16   1895 6.141118e-04
## 17   1896 6.051479e-04
## 18   1897 6.169039e-04
## 19   1898 5.737293e-04
## 20   1899 6.126111e-04
## 21   1900 5.017828e-04
## 22   1901 5.907095e-04
## 23   1902 5.550258e-04
## 24   1903 5.499067e-04
## 25   1904 5.243434e-04
## 26   1905 5.102156e-04
## 27   1906 5.137318e-04
## 28   1907 4.736429e-04
## 29   1908 4.655718e-04
## 30   1909 4.425246e-04
## 31   1910 4.053799e-04
## 32   1911 3.868689e-04
## 33   1912 2.997358e-04
## 34   1913 2.740336e-04
## 35   1914 2.403288e-04
## 36   1915 2.056436e-04
## 37   1916 1.986359e-04
## 38   1917 1.943714e-04
## 39   1918 1.855249e-04
## 40   1919 1.858838e-04
## 41   1920 1.794639e-04
## 42   1921 1.779018e-04
## 43   1922 1.794109e-04
## 44   1923 1.814584e-04
## 45   1924 1.778298e-04
## 46   1925 1.817062e-04
## 47   1926 1.848431e-04
## 48   1927 1.858832e-04
## 49   1928 1.904784e-04
## 50   1929 1.971559e-04
## 51   1930 1.978530e-04
```

```
## 52  1931 2.082921e-04
## 53  1932 2.063539e-04
## 54  1933 2.147613e-04
## 55  1934 2.110177e-04
## 56  1935 2.144944e-04
## 57  1936 2.181519e-04
## 58  1937 2.169611e-04
## 59  1938 2.151304e-04
## 60  1939 2.179173e-04
## 61  1940 2.170693e-04
## 62  1941 2.144655e-04
## 63  1942 2.071141e-04
## 64  1943 2.076156e-04
## 65  1944 2.132730e-04
## 66  1945 2.163425e-04
## 67  1946 2.017719e-04
## 68  1947 1.889952e-04
## 69  1948 1.912796e-04
## 70  1949 1.907812e-04
## 71  1950 1.901058e-04
## 72  1951 1.873256e-04
## 73  1952 1.840656e-04
## 74  1953 1.808511e-04
## 75  1954 1.787660e-04
## 76  1955 1.764225e-04
## 77  1956 1.729511e-04
## 78  1957 1.694996e-04
## 79  1958 1.700200e-04
## 80  1959 1.663820e-04
## 81  1960 1.641475e-04
## 82  1961 1.605969e-04
## 83  1962 1.601109e-04
## 84  1963 1.590282e-04
## 85  1964 1.574027e-04
## 86  1965 1.629578e-04
## 87  1966 1.600521e-04
## 88  1967 1.566016e-04
## 89  1968 1.498546e-04
## 90  1969 1.407308e-04
## 91  1970 1.305796e-04
## 92  1971 1.256902e-04
## 93  1972 1.240797e-04
## 94  1973 1.214376e-04
## 95  1974 1.170620e-04
## 96  1975 1.119282e-04
## 97  1976 1.088652e-04
## 98  1977 1.041740e-04
## 99  1978 1.038398e-04
## 100 1979 9.941684e-05
## 101 1980 9.742368e-05
## 102 1981 9.729416e-05
## 103 1982 9.623298e-05
## 104 1983 9.768859e-05
## 105 1984 9.718731e-05
```

```
## 106 1985 9.424018e-05
## 107 1986 9.142422e-05
## 108 1987 8.806729e-05
## 109 1988 8.415047e-05
## 110 1989 7.908004e-05
## 111 1990 7.599062e-05
## 112 1991 7.469007e-05
## 113 1992 7.359058e-05
## 114 1993 7.190299e-05
## 115 1994 7.169631e-05
## 116 1995 7.137700e-05
## 117 1996 7.037749e-05
## 118 1997 6.878542e-05
## 119 1998 6.645501e-05
## 120 1999 6.489347e-05
## 121 2000 6.215487e-05
## 122 2001 6.105816e-05
## 123 2002 6.050068e-05
## 124 2003 5.934303e-05
## 125 2004 5.769723e-05
## 126 2005 5.681246e-05
## 127 2006 5.417926e-05
## 128 2007 5.278704e-05
## 129 2008 5.255107e-05
## 130 2009 5.307453e-05
## 131 2010 5.400867e-05
## 132 2011 5.433279e-05
## 133 2012 5.460906e-05
## 134 2013 5.547192e-05
## 135 2014 5.565828e-05
## 136 2015 5.593594e-05
## 137 2016 5.612298e-05
## 138 2017 5.689792e-05
```

OUTPUTS:

My outputs are propM which holds each mean for each year without including duplicates. The first item would represent 1880 and so on.

Each_year holds the corresponding year for each mean in propM.

years was a way to access a specific year so I could create a temporary matrix to calculate to mean.

Both of these values were then placed in a dataframe to organize them.

BODY and Sequence:

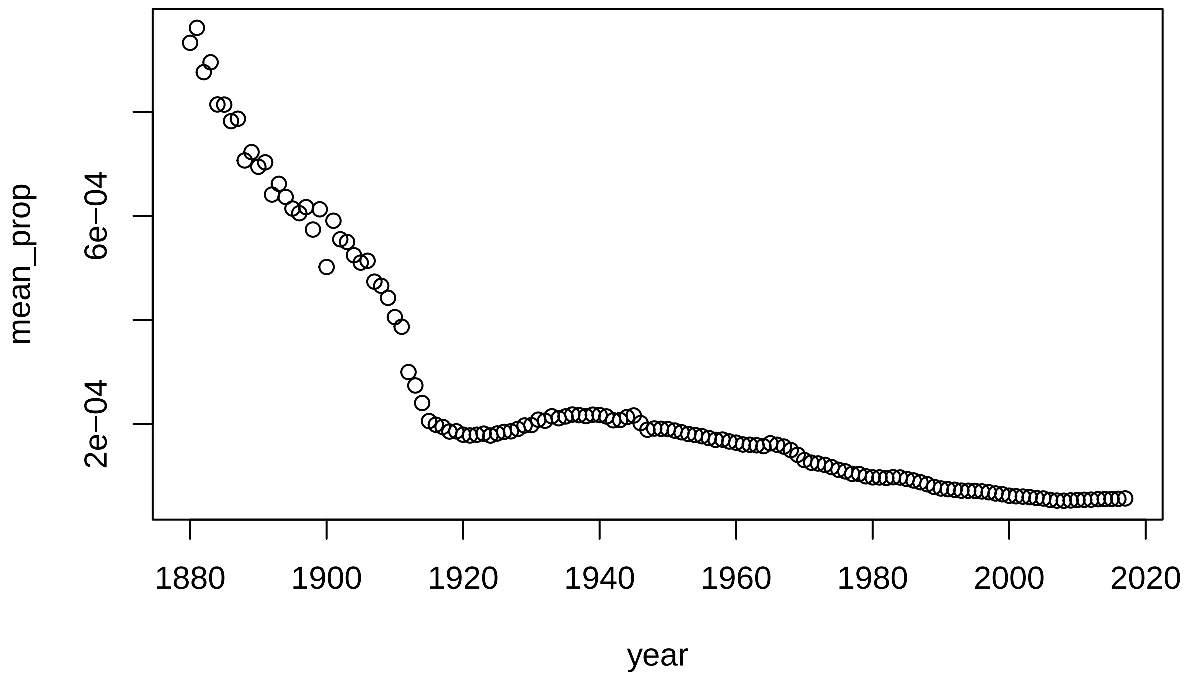
The for loop iterates from 1:138 (length of propM which is how many non duplicate years there are)

It begins by creating a temporary matrix which stores only the data from a particular year (by subsetting). During each iteration, tempMatrix would be reassigned a subset of babynames that only contained the data of a particular year.

I then used the mean() function to calculate the mean prop of the matrix and then added it to prop via indexing. I then also added the current year to Each_year. The loop would repeat this process once it has reached 138/length of prop.

5. Is there a relationship between `year` and its mean `prop`? Make the appropriate graph to show this relationship and add custom axis labels. Describe in words any relationship you see.

```r
with(propM_and_year, plot(year, mean_prop))
```



A: mean_prop = average of proportions of each name in a given year(proportion of people of a particular sex with a particular name) year = each year that this study was conducted.

As the year increases, the prop of names overall decreases each year.A possible conclusion that can be drawn from this data is that people are having less children which is why names are becoming "less popular". If there are less children being had then there would be less names being given, thus decreasing the prop overall each year.

6. Explain why a `for` loop is not the best way to complete the above task. Explore the `tapply` function and use it to produce the same variable you did in question 4. You could use `dplyr` functionality, as well, we just haven't gotten there in class, yet.

```r
tapply(babynames$prop, babynames$year, mean)
```

```
##         1880         1881         1882         1883         1884         1885
## 9.327296e-04 9.615283e-04 8.762491e-04 8.952682e-04 8.141704e-04 8.139021e-04
##         1886         1887         1888         1889         1890         1891
## 7.820985e-04 7.867618e-04 7.065885e-04 7.226397e-04 6.944554e-04 7.028995e-04
##         1892         1893         1894         1895         1896         1897
## 6.410328e-04 6.616759e-04 6.363961e-04 6.141118e-04 6.051479e-04 6.169039e-04
##         1898         1899         1900         1901         1902         1903
## 5.737293e-04 6.126111e-04 5.017828e-04 5.907095e-04 5.550258e-04 5.499067e-04
##         1904         1905         1906         1907         1908         1909
```

```
## 5.243434e-04 5.102156e-04 5.137318e-04 4.736429e-04 4.655718e-04 4.425246e-04
##         1910         1911         1912         1913         1914         1915
## 4.053799e-04 3.868689e-04 2.997358e-04 2.740336e-04 2.403288e-04 2.056436e-04
##         1916         1917         1918         1919         1920         1921
## 1.986359e-04 1.943714e-04 1.855249e-04 1.858838e-04 1.794639e-04 1.779018e-04
##         1922         1923         1924         1925         1926         1927
## 1.794109e-04 1.814584e-04 1.778298e-04 1.817062e-04 1.848431e-04 1.858832e-04
##         1928         1929         1930         1931         1932         1933
## 1.904784e-04 1.971559e-04 1.978530e-04 2.082921e-04 2.063539e-04 2.147613e-04
##         1934         1935         1936         1937         1938         1939
## 2.110177e-04 2.144944e-04 2.181519e-04 2.169611e-04 2.151304e-04 2.179173e-04
##         1940         1941         1942         1943         1944         1945
## 2.170693e-04 2.144655e-04 2.071141e-04 2.076156e-04 2.132730e-04 2.163425e-04
##         1946         1947         1948         1949         1950         1951
## 2.017719e-04 1.889952e-04 1.912796e-04 1.907812e-04 1.901058e-04 1.873256e-04
##         1952         1953         1954         1955         1956         1957
## 1.840656e-04 1.808511e-04 1.787660e-04 1.764225e-04 1.729511e-04 1.694996e-04
##         1958         1959         1960         1961         1962         1963
## 1.700200e-04 1.663820e-04 1.641475e-04 1.605969e-04 1.601109e-04 1.590282e-04
##         1964         1965         1966         1967         1968         1969
## 1.574027e-04 1.629578e-04 1.600521e-04 1.566016e-04 1.498546e-04 1.407308e-04
##         1970         1971         1972         1973         1974         1975
## 1.305796e-04 1.256902e-04 1.240797e-04 1.214376e-04 1.170620e-04 1.119282e-04
##         1976         1977         1978         1979         1980         1981
## 1.088652e-04 1.041740e-04 1.038398e-04 9.941684e-05 9.742368e-05 9.729416e-05
##         1982         1983         1984         1985         1986         1987
## 9.623298e-05 9.768859e-05 9.718731e-05 9.424018e-05 9.142422e-05 8.806729e-05
##         1988         1989         1990         1991         1992         1993
## 8.415047e-05 7.908004e-05 7.599062e-05 7.469007e-05 7.359058e-05 7.190299e-05
##         1994         1995         1996         1997         1998         1999
## 7.169631e-05 7.137700e-05 7.037749e-05 6.878542e-05 6.645501e-05 6.489347e-05
##         2000         2001         2002         2003         2004         2005
## 6.215487e-05 6.105816e-05 6.050068e-05 5.934303e-05 5.769723e-05 5.681246e-05
##         2006         2007         2008         2009         2010         2011
## 5.417926e-05 5.278704e-05 5.255107e-05 5.307453e-05 5.400867e-05 5.433279e-05
##         2012         2013         2014         2015         2016         2017
## 5.460906e-05 5.547192e-05 5.565828e-05 5.593594e-05 5.612298e-05 5.689792e-05
```

```r
#system.time(tapply(babynames$prop, babynames$year, mean)) #about 3 second run time
```

A: For loops in general take longer in R since it is a statistical programming language. In this case, my loop ran faster than tapply function. I think for a much larger data set this will not be the case. Although it ran somewhat faster, it is more complicated than just using vectorize function. There are more variables, and more operations being performed (that are visible) which can make it harder to understand and make it slower when the ranges become larger (e.g 1- 1e^9).

Bonus: Given my kids recent birthday, tell me your best dad joke. Double bonus if it has relevance to this class: